PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/32, 9/42

A1

(11) International Publication Number: WO 98/36350

(43) International Publication Date: 20 August 1998 (20.08.98)

(21) International Application Number: PCT/SE98/00190

(22) International Filing Date: 3 February 1998 (03.02.98)

(30) Priority Data:
9700475-8 12 February 1997 (12.02.97) SE

(71) Applicant *(for all designated States except US)*: TELEFON-AKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: HOLMBERG, Per [SE/SE]; Flintbacken 18, S-118 42 Stockholm (SE).

(74) Agents: SANDSTRÖM, Staffan et al.; Bergenstråhle & Lindvall AB, P.O. Box 17704, S-118 93 Stockholm (SE).

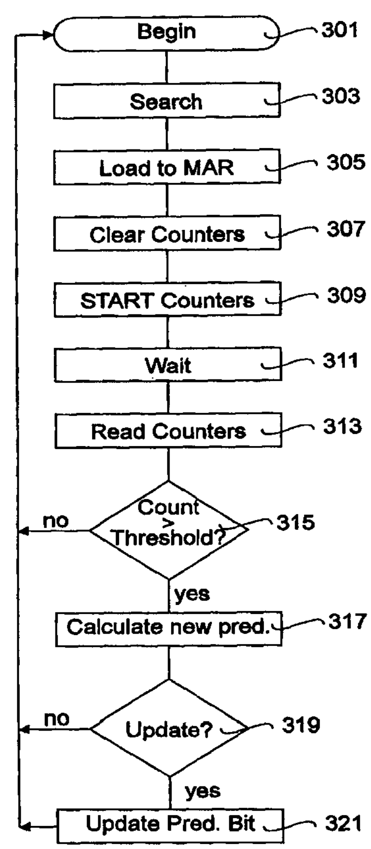
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND SYSTEM FOR BRANCH PREDICTION

(57) Abstract

In a computer a system for branch prediction is arranged. The branch prediction system uses a scanning mechanism (303) for scanning the program memory for conditional branch instructions during the running of the program. When finding such an instruction the system records during a preset time interval (311) the statistics for that specific conditional branch instruction and sets a branch prediction bit in the instruction accordingly (321). The system then starts to scan for the next conditional branch instruction in the program memory. The system can also be used for updating a BHT during the running of a program. The use of the system is particularly useful in applications when a program is run for a relatively long time such as a program used in a telephone switch. The use of the system also allows for changing branch predictions during the run of a program.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND SYSTEM FOR BRANCH PREDICTION

TECHNICAL FIELD

The present invention relates to a method and system for branch prediction in a computer system. The method and the system are particularly well suited for use in processors executing programs running for a long time such as the ones used in telecommunication systems.

BACKGROUND OF THE INVENTION AND PRIOR ART

Branch prediction mechanisms can be loosely divided into static branch prediction and dynamic branch prediction mechanisms.

Static branch prediction is implemented by including a prediction within the branch instruction, i.e. a bit that gives an indication to the processor executing the conditional branch instruction whether a conditional branch is likely to be taken or not. This bit is set by the compiler based on either heuristics, i.e. a conditional branch out of loops is most often not taken, or based on feedback from program execution. The feedback from execution is collected by means of having a program inserting instructions around each conditional branch which records whether the branch is taken or not. The program is then executed and statistics are collected. Thereupon, the program is compiled once again and the collected branch statistics is used to set branch prediction.

Dynamic branch prediction collects branch statistics in separate data structures in the processor, for example in branch history tables, BHTs, or in separate bits in the processor instruction cache or memory. Usually one or two bits in an instruction cache line are used.

The disadvantages with these methods are:

- Setting static branch prediction based on heuristics does not give optimal performance.
- Setting static branch prediction based on feedback gives a number of extra steps in the program generation and works well only as long as the branch statistics collected are similar to real execution in systems using varying and different data sets.

- Dynamic branch prediction adds cost for additional data structures within the CPU. Due to physical limitations, as well as costs, these structures can not include data for all conditional branch instructions in the program and several data branches have to share entries within a BHT. The performance of dynamic branch prediction then depends on the statistical behaviour of the program. For example, if the lower bits of the address of the conditional branch instruction are used to select an entry in the BHT, the performance can depend on whether or not the program has been loaded on addresses that make more than one often executed branch.

In telecommunication applications, programs are loaded into the system and will be used continuously for a long time, i.e. usually at least for weeks, until the system is reloaded with a new revision of the program. The execution can in most cases be expected to have the same statistics during that time.

Furthermore, US 5 367 703 describes a branch prediction mechanism in a superscalar processor system. The mechanism uses branch history tables which include a separate branch history for each fetch position within a multi-instruction access. A prediction field consisting of two bits is used for determining whether a particular branch is to be taken or not. The value of the two bits is incremented or decremented in response to a branch being taken or not.

US 5 423 011 discloses an apparatus consisting of an associated memory in which branch prediction bits are stored, cache lines and comparison means for matching stored prediction bits with their corresponding cache lines.

In the patent application GB 2 283 595 a branch prediction circuitry which can operate in one of the two user selectable modes is described.

SUMMARY

It is an object of the present invention to provide a method and a system which overcomes the problems as outlined above, and

which can provide a branch prediction mechanism which can take advantage of the fact that a program is run for a long time.

This object is obtained with a semi-static branch prediction mechanism comprising three parts:

1) a branch prediction bit in the instruction, or an extra bit in the instruction memory,

2) a hardware counter that can collect branch statistics for a specific conditional branch instruction in the program memory, and

3) a background program.

The branch prediction mechanism then operates as follows:
The background program, e.g. a program having a low priority, a periodic recurrent program, etc., reads the instruction memory to locate conditional branch instructions.

When finding a conditional branch instruction the background program starts the hardware counter to record branch statistics for that branch and goes to sleep for a while.

After waking up, the background program uses the collected statistics to set the prediction in the conditional branch instruction in the program memory.

A system operating in such a manner has several advantages, such as:

- It is transparent for software, and even if the instruction in the program memory is used for storing branch prediction information, there is no impact on any software development tools and the way of storing information can be changed between CPU implementations.
- The hardware design is simple.
- The hardware cost is low, since no separate data structures are needed for branch prediction.

- The method makes it possible to predict multiple conditional branch instructions in parallel, which for example can be needed in superscalar processors for getting good branch prediction accuracy.
- The program performance depends on the execution statistics for the predicted branch only, not on interaction with other conditional branch instructions in other programs.

However, there are some conditions that must be met for the semi-static branch prediction mechanism to work well, i.e. to provide a good branch prediction.

i) The same program must be executed for a relatively long time and have approximately the same behaviour during that time since it takes some time for the background program to scan the program for all conditional branch instructions, and collect reliable statistics.

ii) It must be possible to include the branch prediction bit.

iii) It must be possible to include a hardware counter or counters for collecting execution statistics.

The APZ processors used in the AXE telephone switch manufactured by Ericsson fulfil all these requirements. The hardware and software are custom designed and most conditional branch instructions have several unused bits, which can be used for storing branch prediction.

Furthermore, the background program and the counters can be used for updating a branch history table (BHT). Instead of updating the prediction bit in a conditional branch instruction after each time new statistics are collected, the background program is used for updating the prediction field corresponding to the instruction in the BHT.

Such an implementation can, for example, be advantageous when it is not possible to include the branch prediction bit.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described in more detail by way of non-limiting examples and with reference to the accompanying drawings, in which:

- Fig 1 is a general view of a unit comprising parts of a computer involved in a branch prediction mechanism supplemented with hardware and software for performing semi-static branch prediction.
- Fig 2 is a detailed view of the counter hardware used by the unit in fig. 1.
- Figs. 3a and 3b are flow charts used in a background program used for setting a branch prediction bit and for updating a BHT, respectively, in a branch prediction mechanism.

DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 illustrates a unit 101 built of a number of blocks which are usually involved in a branch prediction mechanism. Thus, the unit 101 has a program memory 103 in which the program to be executed by the processor is stored. The program memory 103 is usually connected to a cache memory 105. However, the use of the cache memory 105 is optional.

The program memory 103 or the cache memory 105, if such a one is used, is connected to a memory interface 107. The object of the interface 107 is to provide an interface between the memory and an instruction decode block 109. Thus, the block 107 is used for fetching instructions from the memory which then are provided to the instruction decode block 109.

In the instruction decode block 109, the instruction is decoded. When the instruction has been decoded, the processor has knowledge of the type of instruction, which currently is processed. This information is used to evaluate if the instruction is a conditional jump instruction or not. The information if the instruction is a jump instruction or not, is fed to an instruction fetch unit in a block 111 together with information on the address to which the possible jump goes from the block 109. The block 111 also comprises a branch predictor

setting means 119.

The information on the address to which the possible jump goes can be fed in various manners such as by means of providing the absolute address directly or as an address relative to the present address, i.e. a relative address. Another way of indicating the address to which a possible jump goes is to provide a parameter. The parameter is then used as an entry to a table which then outputs the address. The latter method is used in the APZ processor developed and manufactured by Ericsson.

The instruction fetch unit in the block 111 then fetches the next instruction based on the information provided from the branch predictor 119. Thus, if the prediction information provided from the branch predictor setting means 119 indicates that the current instruction is a conditional jump instruction, and the jump is decided to be likely to be taken in the block 111, the instruction at the address indicated by the prediction information from the branch predictor setting means 119 is selected to be fetched next.

If, on the other hand, the information from the branch predictor 119 indicates that the current instruction was not a conditional jump instruction, or if the block 111 decides that the jump is not likely to be taken, the instruction at the next sequential instruction address is chosen to be fetched.

The block 111 is also connected to the program memory, and possibly also to the cache memory 105 in order for a unit for collecting statistics 121 located therein to update prediction bits in the memories 103 and 105.

The instruction decoded in the block 109 is then further processed in an execution unit. Usually a processor, as in this case, has several execution units each designed for executing different types of instructions. Hence, the unit 101 is equipped with three execution units 113, 115 and 117. The first unit 113 is used for executing instructions involving integer operations, the second unit 115 is used for executing instructions involving

floating point operations and the third execution unit 117, the branch unit, is used for executing jump instructions.

Thus, depending on the type of instruction which is to be executed the decoded instruction from the block 109 is fed to one of the three execution units in blocks 113, 115 or 117.

In the branch execution unit in block 117 information on the outcome of each conditional jump instruction is recorded. This is performed by means of collecting information from the other two execution units in the blocks 113 and 115. When the branch unit in block 117 has collected all information required for evaluating both if a conditional jump was carried out and, if so, to which address the jump went, this information is fed to the block 111. The block 111 uses the feedback information from the block 117 when determining the address from which the next instruction is to be fetched. Thus, if a previous conditional jump has been mispredicted the correct instruction at the correct address must be fetched and instructions fetched from the misprediction and onwards must be ignored.

In fig. 2 the hardware used in the unit 121 for collecting statistics regarding if a branch is taken or not, is shown. Thus, for collecting statistics regarding a certain conditional branch instruction in the program memory, the address of that instruction is placed in a register 201, here termed Measured Address Register (MAR). This address is compared in a block 203 with the instruction address currently pointed to by the program counter and which is available in a block 205.

The two addresses are compared in the block 203 and if the two addresses are identical a first counter in a block 211 is incremented by one. The output from the block 203 is also fed to an AND block 207. To the AND block 207, a signal indicating if the branch was taken or not is also fed. Thus, the output from the block 207 increments a second counter 209 each time the branch in the instruction in the memory address register is taken.

In general, two out of the following statistics counts need to be collected for setting the branch prediction bits:

- the number of times the conditional branch is taken
- the number of times the conditional branch is not taken
- the total number of times the conditional branch instruction is executed.

Figs. 3a and 3b are flow charts illustrating a background program used for collecting statistics regarding different conditional jump instructions and for setting prediction bits accordingly. The counters used by the program are those described in conjunction with fig. 2.

Thus, the background program begins with scanning or searching the program memory for the first conditional jump instruction in a block 303. When finding the first conditional branch instruction the corresponding program memory address is loaded into the Measured Address Register (MAR) in a block 305. Thereupon the program clears all counters used for collecting the statistics in a block 307.

Next, all counters are started in a block 309. The background program now waits for statistics to be collected. The counters are incremented each time the program from which statistics are collected executes the conditional branch instruction associated with the address stored in the MAR and when the corresponding branch is taken, respectively, if the implementation as described in conjunction with fig. 2 is used. The statistics for a specific conditional branch instruction are collected for a predefined time as indicated in block 311, which can be equally long for each conditional branck instruction.

Thereafter, the counters are read in a block 313. If the conditional branch instruction was executed very few times during the measurement period the background program returns to the block 303. This is determined in a block 315 for example by means of comparing the number of times the conditional branch instruction was executed to a preset threshold value. If, on the

other hand the number of times the conditional branch instruction was executed is large enough for assuring relevant statistics the background program continues to a block 317.

In the block 317, the new prediction is calculated. The background program then proceeds to a block 319. In the block 319, it is decided if the branch prediction bit is to be updated or not.

Thus, if the number of times the conditional branch was taken and not taken, respectively, were equal or almost equal, the decision is no, and the background program returns to the block 303. If, on the other hand, the decision is yes the background program proceeds to a block 321.

In the block 321 the prediction bit in the conditional branch instruction is updated in the program memory and possibly also in the cache memory if used. The background program then returns to the block 303 in which the search for a next conditional branch instructions begins, or if the instruction was the last conditional branch instruction in the memory the background program starts scanning from the beginning of the program in the program memory.

The method can thus be used to either update an extra bit in the instruction memory or a branch prediction bit in the instruction.

In another preferred embodiment the statistics collected by the background program are used for updating a branch history table (BHT). Thus, in such an embodiment, instead of changing a prediction bit in a conditional branch instruction, the background program is used for changing the BHT.

The flow chart for such an implementation can be identical to the flow chart in fig. 3a except that the block 321 is replaced by a block 323 in which an update of the BHT is performed instead. In fig. 3b the flow chart for such an implementation is shown.

The use of a system changing a BHT instead of a branch prediction bit in the instructions can be advantageous in cases when a prediction bit is not available in the conditional branch instructions or if the prediction system as described herein is applied in a computer already using a BHT. In the latter case very little extra hardware and software need to be added.

CLAIMS

1. A system for branch prediction, **characterized by**
- means for recording during a time interval the number of times a specific branch of a conditional branch instruction in a running program is taken or not taken, respectively, and
- means connected to the recording means for setting a branch prediction bit in the instruction, or an extra bit in the instruction memory, to a value depending on the number of times the branch in the instruction was taken or not taken during the recording time.

2. A system according to claim 1, **characterized by** means for scanning the running program for another conditional branch instruction and starting to record during a new time interval when a preceding time interval has elapsed.

3. A system according to claim 2, **characterized in** that all time intervals are equally long and preset.

4. A system according to any of claims 1 - 3, **characterized in** that the means for setting the branch prediction bit is arranged to set the branch prediction bit in the branch instruction if the branch is taken more times than it is not taken during the recording interval and otherwise to reset the branch prediction bit.

5. A system according to any of claims 2 - 4, **characterized by** means for increasing the recording time interval for a specific branch instruction, if during a last recording for that specific conditional branch instruction the number of total recorded executed instructions was below a preset threshold value.

6. A system according to any of claims 2 - 4, **characterized by** means for not changing the branch prediction bit, if at the end of a recording the number of total recorded executed instructions is below a preset threshold value, regardless of the outcome of the collected statistics.

7. A computer comprising:
- a program memory for storing a computer program,
- a decode unit for decoding instructions in a computer program stored in the memory,
- an interface unit connected to the memory, and to the decode unit, for fetching instructions from the memory to the decode unit, and
- an execution unit for executing the instructions decoded by the decode unit,

characterized by
- means connected to the decode unit and to the execution unit for recording during a time interval the number of times a specific branch of an instruction in a running program is taken or not taken, respectively and
- means connected to the memory and to the recording means for setting a branch prediction bit in the instruction, or an extra bit in the instruction memory to a value corresponding to the number of times the branch in the instruction was taken or not taken during the recording time.

8. A computer according to claim 7, **characterized by** means for setting the branch prediction bit to indicate that the branch is to be taken, if the number of times the branch is taken during the recording interval exceeds the number of times the branch is not taken and vice versa.

9. A method for branch prediction, **characterized by** the steps of
- recording during a time interval the number of times a specific branch of a conditional branch instruction in a running program is taken or not taken respectively, and
- setting a branch prediction bit in the instruction or an extra bit in the instruction memory in the running program to a value depending on the number of times the branch in the instruction was taken or not taken during the recording time.

10. A method according to claim 9, **characterized in** that, when a preceding time interval has elapsed, the program is scanned for another conditional branch instruction and that a recording during a new time interval is started.

11. A method according to claim 10, **characterized in** that all time intervals are set equally long.

12. A method according to any of claims 9 - 11, **characterized in** that the branch prediction bit in the conditional branch instruction is set if the branch in the recorded conditional branch instruction is taken more times than it is not taken.

13. A method according to any of claims 9 - 12, **characterized in** that, if during a last recording for a specific conditional branch instruction the number of total recorded executed instructions was below a preset threshold value, the recording time interval for that specific branch instruction is increased.

14. A method according to any of claims 10 - 13, **characterized in** that, if at the end of a recording the number of total recorded executed instructions is below a preset threshold value, the branch prediction bit, regardless of the outcome of the collected statistics, is not changed.

15. A system for branch prediction, **characterized by**
- means for recording during a time interval the number of times a specific branch of a conditional branch instruction in a running program is taken or not taken, respectively, and
- means connected to the recording means for setting an entry of the branch history table (BHT) corresponding to the address of the recorded instruction to a value depending on the number of times the branch in the instruction was taken or not taken during the recording time.

16. A computer comprising:
- a program memory for storing a computer program,
- a decode unit for decoding instructions in a computer program stored in the memory,
- an interface unit connected to the memory, and to a decoding unit, for fetching instructions from the memory to the decode unit, and
- an execution unit for executing the instructions decoded by the decode unit,

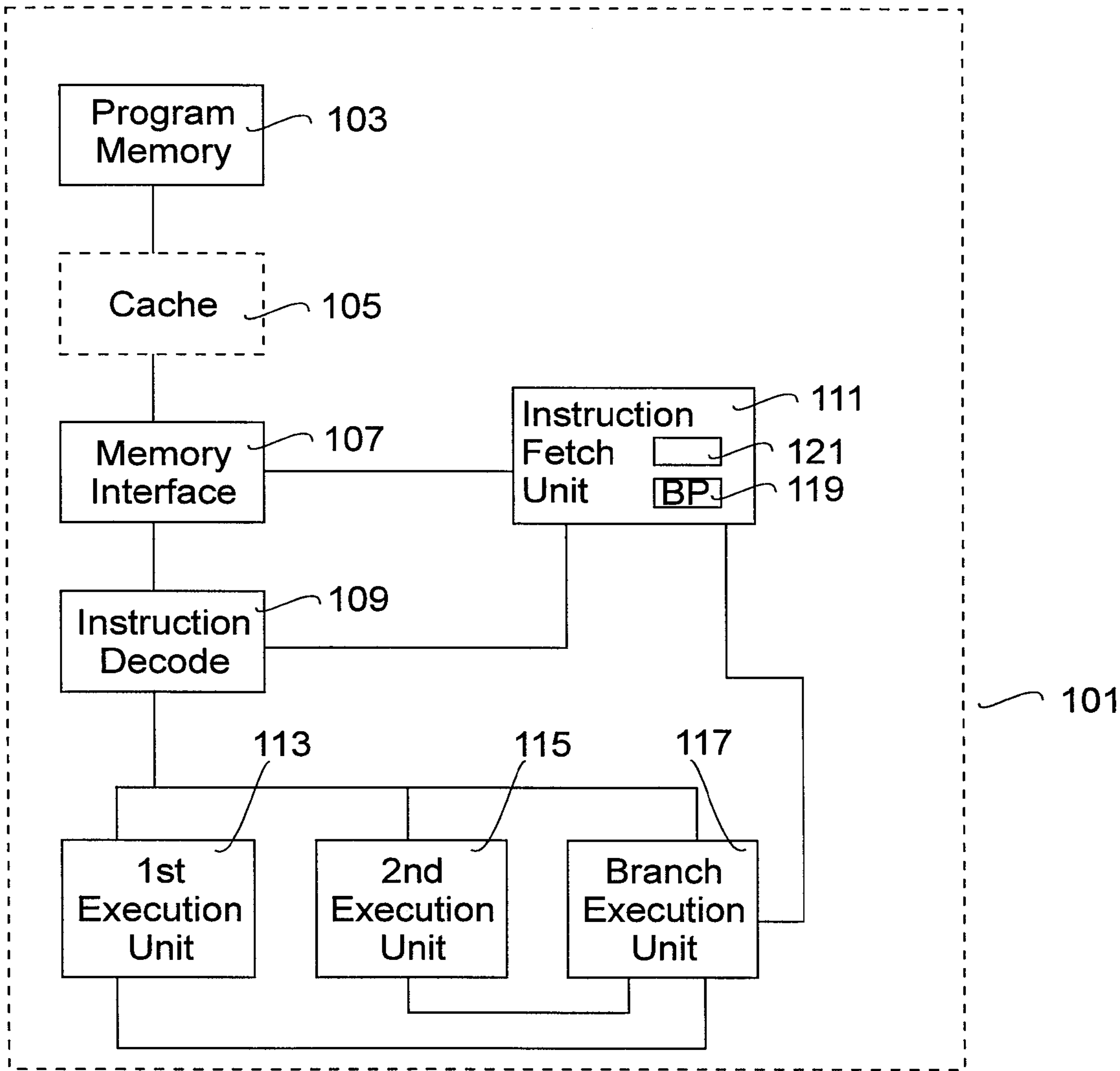

Program Memory
103
Cache
105
Memory Interface
107
Instruction Fetch Unit
111
121
BP
119
Instruction Decode
109
101
113
115
117
1st Execution Unit
2nd Execution Unit
Branch Execution Unit

Fig. 1

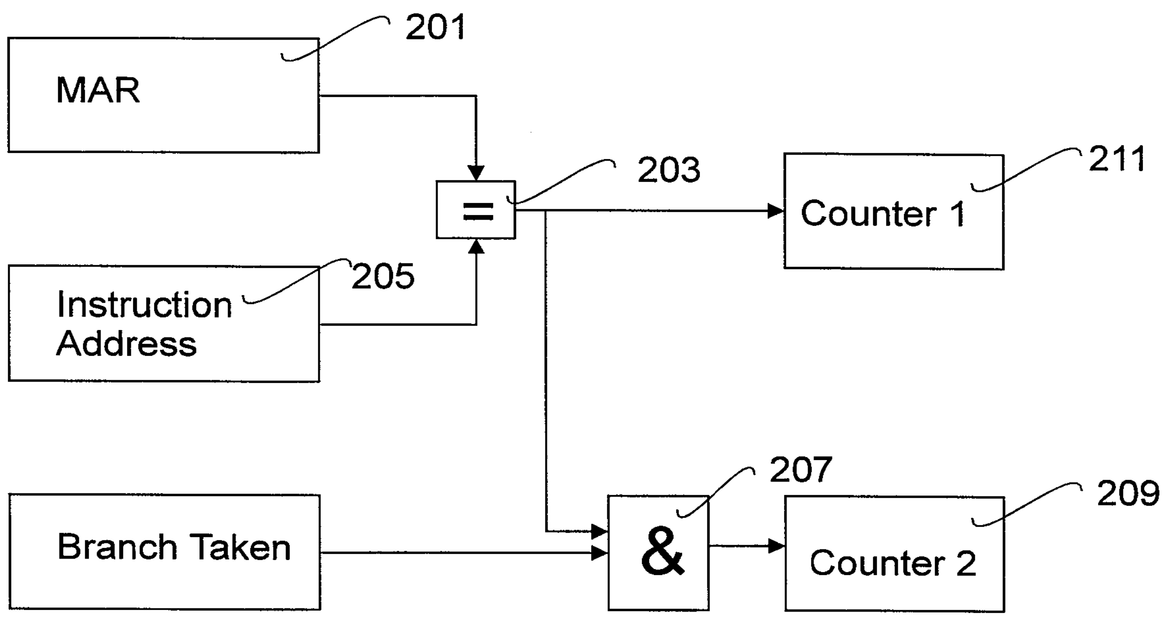

201
MAR
203
Counter 1
211
=
Instruction Address
205
207
209
Branch Taken
&
Counter 2

Fig. 2

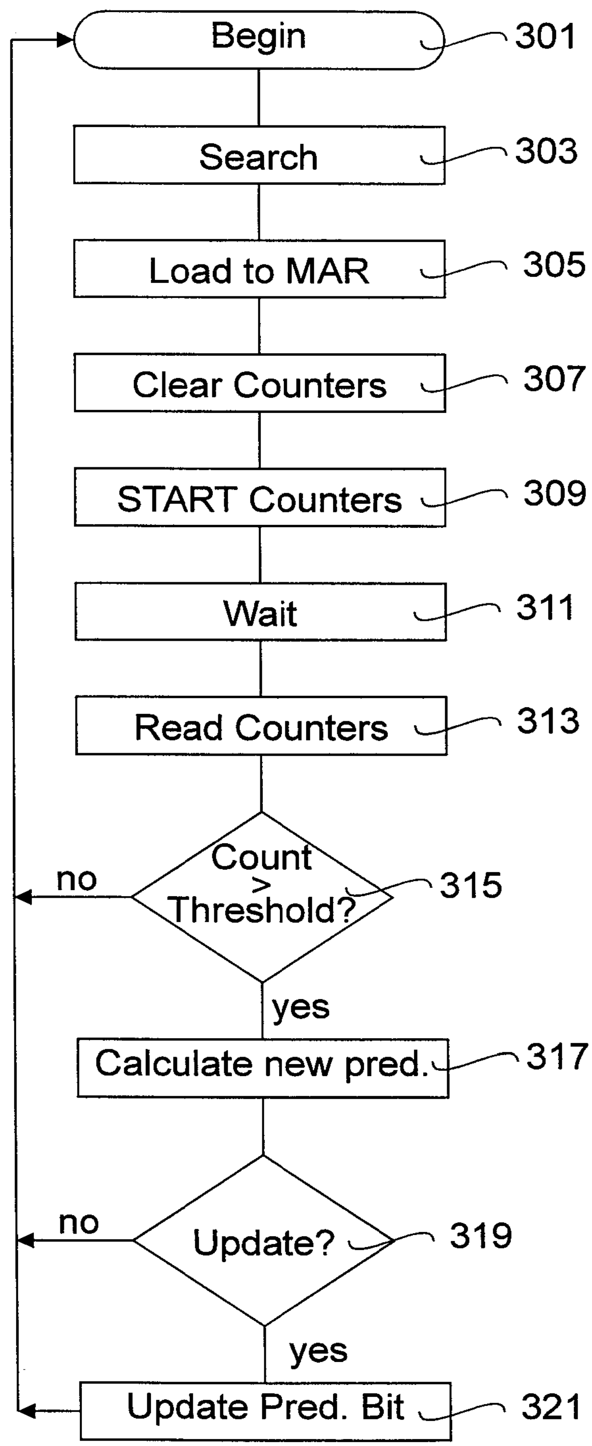

Begin
301
Search
303
Load to MAR
305
Clear Counters
307
START Counters
309
Wait
311
Read Counters
313
no
Count > Threshold?
315
yes
Calculate new pred.
317
no
Update?
319
yes
Update Pred. Bit
321

Fig. 3a

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 98/00190

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 9/32, G06F 9/42
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	SIGPLAN notice, Volume 29, No 11, 1994, Cliff Young et al, "Improving the Accuracy of Static Branch Prediction Using Branch Correlation", page 233, figure 2, paragraph 2 --	1-18
Y	SIGPLAN notice, Volume 27, No 9, 1992, Shien-Tai Pan et al, "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation", page 77, paragraph 2.1 --	1-18
X	US 4334268 A (JOEL F. BONEY ET AL), 8 June 1982 (08.06.82), column 3, line 26 - line 47 --	7

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 June 1998	03-07-1998

Name and mailing address of the ISA/	Authorized officer
Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Jan Silfverling Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

Information on patent family members

09/06/98

International application No.
PCT/SE 98/00190

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4334268 A	08/06/82	NONE	
US 5051944 A	24/09/91	NONE	
US 5394529 A	28/02/95	CA 2045827 A	30/12/91
		EP 0463965 A	02/01/92
		JP 6059887 A	04/03/94
		US 5450555 A	12/09/95
		US 5488730 A	30/01/96
GB 2283595 A	10/05/95	CN 1117618 A	28/02/96
		GB 9420379 D	00/00/00
		IE 940854 A	03/05/95
US 5367703 A	22/11/94	EP 0605876 A	13/07/94
		JP 2531495 B	04/09/96
		JP 6236270 A	23/08/94
US 5440704 A	08/08/95	JP 2033999 C	19/03/96
		JP 7054459 B	07/06/95
		JP 63055639 A	10/03/88
		JP 2034001 C	19/03/96
		JP 7058463 B	21/06/95
		JP 63059630 A	15/03/88